**Mestra - Controller**

Design

Michel Keijzers, © 2017

# History

Table : History

|  |  |
| --- | --- |
| **Date/period** | **Actions** |
| Nov 8, 2017 | Initial version |

# Table of Contents

Contents

[History 1](#_Toc498126522)

[Table of Contents 2](#_Toc498126523)

[List of Tables 2](#_Toc498126524)

[List of Figures 2](#_Toc498126525)

[1 Introduction 3](#_Toc498126526)

[2 Folder Structure 4](#_Toc498126527)

# List of Tables

[Table 1: History 1](#_Toc498126528)

# List of Figures

# Introduction

This document describes the design of the Separate Controller (i.e. controller which do not have any slave functions inside).

It handles:

* Requirements (software and hardware)
* Design (software and hardware)

# Requirements

## Generic

None

## Hardware

Table 18: Requirements CH

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Base ID** | **Version** | **Category** | **Item** | **Description** |
| ConH1 | Gen1, GenS1,  GenS10, GenS20, | 1.0 | External Memory | Amount | The memory size should be high enough to store all rules (triggers/commands).  *Rationale: To meet REQ Gen1, data cannot be loaded from a slow component (SD card).* |
| ConH2 | Gen2 | 1.0 | SD | Amount | The time to copy all rules to external memory should take less than 5 seconds.  *Rationale: another 5 seconds is left for software initialization and device (wireless?) coupling.* |
| ConH10 | Gen10 | 1.0 | Diagnostics | Error | Whenever an error occurs, LEDs will be used to show the root cause. |

## Software

Table 24: Requirements CS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Base ID** | **Version** | **Category** | **Item** | **Description** |
| ConS1 | GenH10 | 1.0 | Flash | Amount | The amount of Flash memory is limited.  *Rationale: Cost perspective.* |
| ConS10 |  | 1.0 | SRAM | Usage | The total combination of triggers, rules and commands are only limited by the storage (there should be no predefined mapping).  *Rationale: More flexibility* |

# Hardware

## Inputs/Outputs

The controller will have the following inputs:

* USB, only used to flash the sketch.
* Adapter, for power.
* SD card, for the SD card containing the configuration file.
* Wifi (wireless), for wireless transmission of the sketch.
* RF (wireless), for receiving messages from slave devices.

The controller will have the following outputs:

* RF (wireless), for sending messages to slave devices.
* LEDs, for receiving/transmission and diagnostics information.

## Design

### Arduino Type

Due to REQ ConH20, 2 KB for an Arduino Uno is on the low side, therefore an Arduino Mega is chosen.

Also, during development, it is comfortable to have the RX available for debugging, and having multiple UARTs is convenient.

Most important, the Arduino Mega has 256 KB flash instead of 16 KB. It is expected, the implementation of all commands, triggers and rules may be consuming way more than 16 KB.

### External memory selection

Excel sheet <TODO> shows 128 KB is enough to store everything according REQ ConH1.

An Arduino Uno has 2 KB, an Arduino Mega 4 KB internal memory. Therefore, external memory is needed.

Two types of memory are taken into account:

Table 19: Comparison External Memory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **3K256 SRAM** | **23LC1024 SRAM** | **LP16040 (TODO)** | **EEPROM** |
| Reading speed | +++ | +++ | - | +++ |
| Writing speed | +++ | +++ | + | - |
| Wear-off | +++ | +++ | +++ | --- |
| Ease of use | -- | +++ | --- | + |

The reading speed of LP16040 is low, since it is a DIP24 IC, which needs either lots of digital pins, or a shift register, resulting in low read/write times.

EEPROM has problems when needing to write often that is wears off the IC.

3K256 SRAM has only 32 KB and 4 ICs are needed, with logic to combine them.

Therefore, it is clear a 23LC1024 is the best solution.

### SD Card

Due to CH210, the SD card should be copied to external memory within seconds.

The typical SD card speed using SPI on an Arduino is like 100 KB/s, reading 128 KB results in approximately 1 second, which is well within spec.

### Communication between external application and PC

The configuration file generated from the external application needs to be stored on the controller. This can be done in multiple ways:

Table 20: Comparison Communication Ext Application/PC

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **SD** | **USB cable** | **Wireless** |
| Ease of use | -- | + | +++ |
| Extra hardware needed | +++ | +++ | --- |
| Enclosure complexity | -- | +++ | + |
| Reliability | +++ | +++ | + |
| Usability | +++ | +++ | -- |
| Software complexity | +++ | --- | --- |

It shows a USB cable is the best option. However, making a SD opening in the enclosure of the controller, makes it possible to use both the SD and USB cable option.

The wireless option will be not implemented, since it might well be possible a wireless network is present (although very likely during creating the configuration file). But except the transferring of the configuration file, it has no further use. Also using an RX/TX signal, it might be problematic when using RX/TX for another wireless solution to/from the slaves (see next paragraph).

### Diagnostics LEDs

Table 21: Controller Diagnostics LEDs

|  |  |  |
| --- | --- | --- |
| **Function** | **LED Color** | **Description** |
| Power | Blue (generic) | Off: Power off  On: Power on |
| RF, one per slave (max 5) | Yellow (generic) | Off: empty message transmitting/receiving  Slow blinking: contact with controller  Double fast blinking per second: no contact with slave  Triple fast blinking per second: problem with RF  On: non empty message transmitting/receiving |

Note that if the GUI Device shows errors whenever possible.

### Diagnostics/Errors

Available LEDs:

* On/Off (blue)
* Receiving (green)
* Sending (red)

Whenever an error occurs, LEDs will display as shown below.

Table 22: Diagnostics Controller

|  |  |  |
| --- | --- | --- |
| **Root Cause** | **LEDs** | **Solutions** |
| SD Card cannot be read | On/Off: Blinking  Receiving: Off  Sending: Off | Reinsert the SD card and repower the device. |
| SRAM not accessible | On/Off: Blinking  Receiving: Off  Sending: On | None. |
| RF not accessible | On/Off: Blinking  Receiving: On  Sending: Off | None. |

Note that if the GUI Device shows errors whenever possible.

## Breadboard Layout

The breadboard will contain the RF communication breakout board, the adapter for it, and the SRAM chip including LEDs. This will take up maximum a half bread board.

## Proto Layout

TODO

## Component List

The cost of a prototype will be approximately (see Excel document, tab Cost):

Table 23: Components Controller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Item Cost** | **Amount** | **Total** | **Datasheet** |
| Arduino Mega | € 7,35 | 1 | € 7,35 |  |
| SD Recorder Shield | € 1,50 | 1 | € 1,50 |  |
| SD Card 128 MB | € 3,00 | 1 | € 3,00 |  |
| Enclosure | € 5,00 | 1 | € 5,00 |  |
| SRAM | € 3,50 | 1 | € 3,50 |  |
| 2 x RF Transceiver | € 0,80 | 2 | € 1,60 |  |
| Various electronics | € 2,00 | 1 | € 2,00 |  |
| Enclosure | € 5,00 | 1 | € 5,00 |  |
| **Total** |  |  | **€ 28,95** |  |

# Software

## Design

The controller has by far the most extensive sketch. The reason for this is, that all intelligence is built into this device. It receives messages, transforms them, and sends them to the correct slave(s).

## Memory Usage

### Introduction

A lot of information need to be stored. Because of REQ ConS10, the default 8 KB SRAM of the Arduino Mega is not enough.

The internal 8 KB SRAM will be used for only the highest necessary data (REQ ConS1)

* Stack trace (can be larger than other modules, due to the complexity of rules and the OO kind of programming)
* Buffers (initially 512 bytes for the SD card -> SRAM buffer, these buffers will be removed after copying).
* Local variables

Therefore, none other lists should be saved in SRAM. All lists will be stored in external SRAM (unless REQ Gen1 is affected).

The external 128 KB SRAM will be used for storing the rules and commands for all (slave) devices. To be most flexible, the SRAM will not be divided evenly per slave, but used as needed.

Also, memory consumption should be kept low.

In the following paragraphs, per slave device the memory layout/usage is explained.

### Trigger filtering

The Controller will send messages to the slaves for which signals messages will be sent towards the Controller. This way the Controller will only get messages that really needs to be processed, reducing the number of packages to be sent by the RF network and keeping the processor of both Controller and slaves lower.

What needs to be stored are all trigger tables themselves, and all commands and all current states (because commands rely on them). Below per slave the memory usage/mapping will be explained.

### Generic

TODO

### MIDI

### Pedals/switches

Assuming there will be 8 pedals and 8 switches, the size needed is 8 (switches) \* 2 (type on/off) \* 2 (address size) + 8 (pedals) \* 3 (types) \* 2 (address size) = 32 + 48 = 80 bytes, which is negligible.

## Timing Performance

TODO

# Testing

## Unit Tests

The Controller has the most complex software thus testing is critical.

The classes regarding rules, commands, triggers etc. are not using Arduino specific code. This means the tests can be created by running them on a standard PC.

## Integration Tests

The following tests will be performed (by connecting a keyboard/synthesizer):

1. Play a note, same note should be played.
2. Play many notes, all notes should be played, no notes should be heard after releasing all notes.
3. As 1, while moving the joystick.
4. As 2, while moving the joystick
5. As 1, while using aftertouch
6. As 2, while using aftertouch
7. Specific tests for rules, commands etc. TODO

RF Interference tests:

1. Testing with one or more mobile phones communicating via WIFI
2. Testing with one or more mobile phones communicating via Bluetooth
3. Testing while one or more mobile phones are starting to communicate via WIFI
4. Testing while one or more mobile phones are starting to communicate via Bluetooth
5. Putting a microwave near the controller or slave devices